The Terabyte Analysis Machine Project

The Distance Machine: Performance Report

James Annis, Gabriele Garzoglio, Chris Stoughton (Fermilab)
Koen Holtman (Caltech), Peter Kuntz (JHU)

Abstract: The Terabyte Analysis Machine Project is developing hardware and software to analyze Terabyte scale datasets. The Distance Machine framework provides facilities to flexibly interface application specific indexing and partitioning algorithms to large scientific databases.

Overview

The Sloan Digital Sky Survey is the first of the new generation of large data volume sky surveys that will generate tens of Terabytes of images and put billions objects into catalogs. To deal with the analysis of Terabyte scale SDSS data sets, we are converging on Datawolf design concepts: if Beowulfs are clusters optimized for efficient message passing parallel processing, Datawolves are clusters optimized for both high-I/O-rate scans through large data sets and for bringing high compute power to bear on large datasets. The datawolf design necessarily involves sophisticated database software and multidimensional search algorithms. Our prototype datawolf is the Terabyte Analysis Machine [1], a cluster of dual CPU Linux boxes, with large amounts of memory and local disk and access to a Terabyte of raid disk via a SAN (a fibre channel network and Global File System), hosting a copy of the SDSS database SX [2]

We report on the analysis engine framework. The main idea is to apply to specific science problems sophisticated computer science algorithms. Since the interesting astronomy questions often devolve to the act of measuring the distance from one object to every other object, for all objects, we named the framework the Distance Machine. At its heart are algorithm libraries. We aim to make it straightforward for a scientist to insert an algorithm library. Our prototype analysis is a search for clusters of galaxies, and we chose as the algorithm the determination of the k^{th} nearest neighbors. We modified and extended an existing library, ANN (Approximate Nearest Neighbors [3]), which provides methods to find in memory the k^{th} nearest. The central algorithm of ANN is the kd-tree.

The prototype is a layer of middleware between the SX and the user: it extracts the subset of useful objects from SX, builds a persistent kd-tree in Objectivity using the k^{th} nearest neighbor algorithms, and presents server-client based CORBA interface to the science analysis code. Transparent access to the large database embodied dataset is granted by embedding the Objectivity reference to each data object within a wrapper, a wrapper whose standard array operators are overloaded. A vector of wrappers can be effectively treated as a vector of pointers to memory: this approach not only allows almost immediate extension of most existing standalone analysis applications to large Objectivity managed datasets, but also enables new applications to use the re-indexing and re-partitioning features of the framework to increase the efficiency of the data access.

Building and searching the tree

First, ANN/Objectivity loads each objectivity data reference into a vector of wrappers whose role it is to overload the standard array operators. For our database configuration (1 container per database, max test database size 200 Megabytes, default ooinit() parameters, object size 250 bytes) this takes 7 μs/object. Then the user provided definition of the parameter space (schema + overloading rule for operator[]) is used to build in memory a tree structure on the data. At each node of the tree (hyper-cubes in the parameter space) summary information like node boundaries are maintained. Pointers to the data wrappers are stored in the leaf nodes. The tree structure can be saved in a text format for future retrieval: for our system, the retrieval time is of the order of 10 μs/object, while building the tree is a CPU intensive operation of the order of fraction of ms/object. The data agree with the expected $O(n \log(n))$ complexity of the algorithm. The search for a NN is made extremely quick using the summary meta-data maintained in the nodes of the tree. Fig.1 shows a comparison between the standard $O(n^2)$ algorithm and the tree based algorithms that ANN implements.

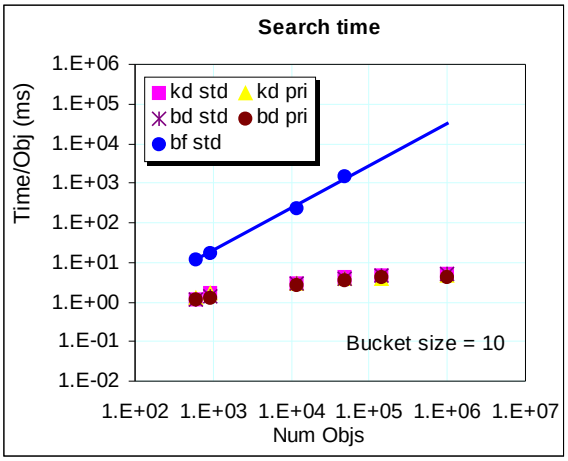

Fig 1: Comparison between the search time of the standard $O(n^2)$ NN algorithm (bf std) and the tree based ones, kd-tree and bd (box decompositon) tree. Two different search algorithms are used for the trees: std and pri (priority queue).

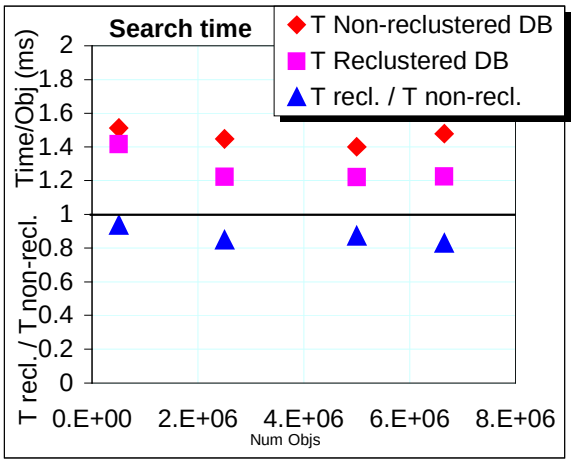

Fig 2: Comparison in the search time per object for non-reclustered and reclustered databases, for different database sizes.

The re-clustering module

The process of organizing the data in a tree structure introduces a natural indexing on the data. The data can be reorganized on disk according to this indexing (re-clustering). On our system, this process takes about 80 μs/object. This technique aims to minimize the amount of memory cache swaps needed by objectivity during the search for the nearest neighbors. Fig. 2 compares the time per object required to search 5 nearest neighbors for all the objects, in the two cases of standard and re-clustered database. For a small database (500,000 objects) the time is approximately the same: the data size in this case is small (160MB) and few memory swaps are needed; for bigger databases (above 2,500,000 objects) there is a gain of 15-20% in the query execution. In general the gain can be better, considering that in this case the initial database was already partially clustered in two (ra, dec) of the 4 dimensions used (spatial organization).

The client/server architecture.

ANN/Objectivity is currently implemented as a C++ library. To allow flexibility in the choice of user analysis programming language, a client/server architecture implemented via Corba interfaces has been provided. The search time per object for non-reclustered databases increases to less than 1.8 ms/object. This result has to be compared with 1.4 ms/object of Fig 2: the overhead is acceptable in most cases, where the increased flexibility allows rapid development and data exploration.

References

[1] Annis, J., Garzoglio, G., Ruthsmandorfer, K., Stoughton, C., "Terabyte Analysis Machine" Advanced Computing and Analysis Techniques in Physics Research 2000, ed. Bhat and Kasemann., www-sdss.fnal.gov:8000/~annis/Tam/tam.ps

[2] Szalay, A., Kunszt, P., Thakar, A., Gray, J., Slutz, D., "The Soan Digital Sky Survey and its Archive", Astronomical Data Analysis Software and Systems IX, 1999

[3] ANN Programming Manual; David M. Mount, Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, www.cs.umd.edu/~mount/ANN/